Technical Cybersecurity

More with f2

Execution

Break in our functions

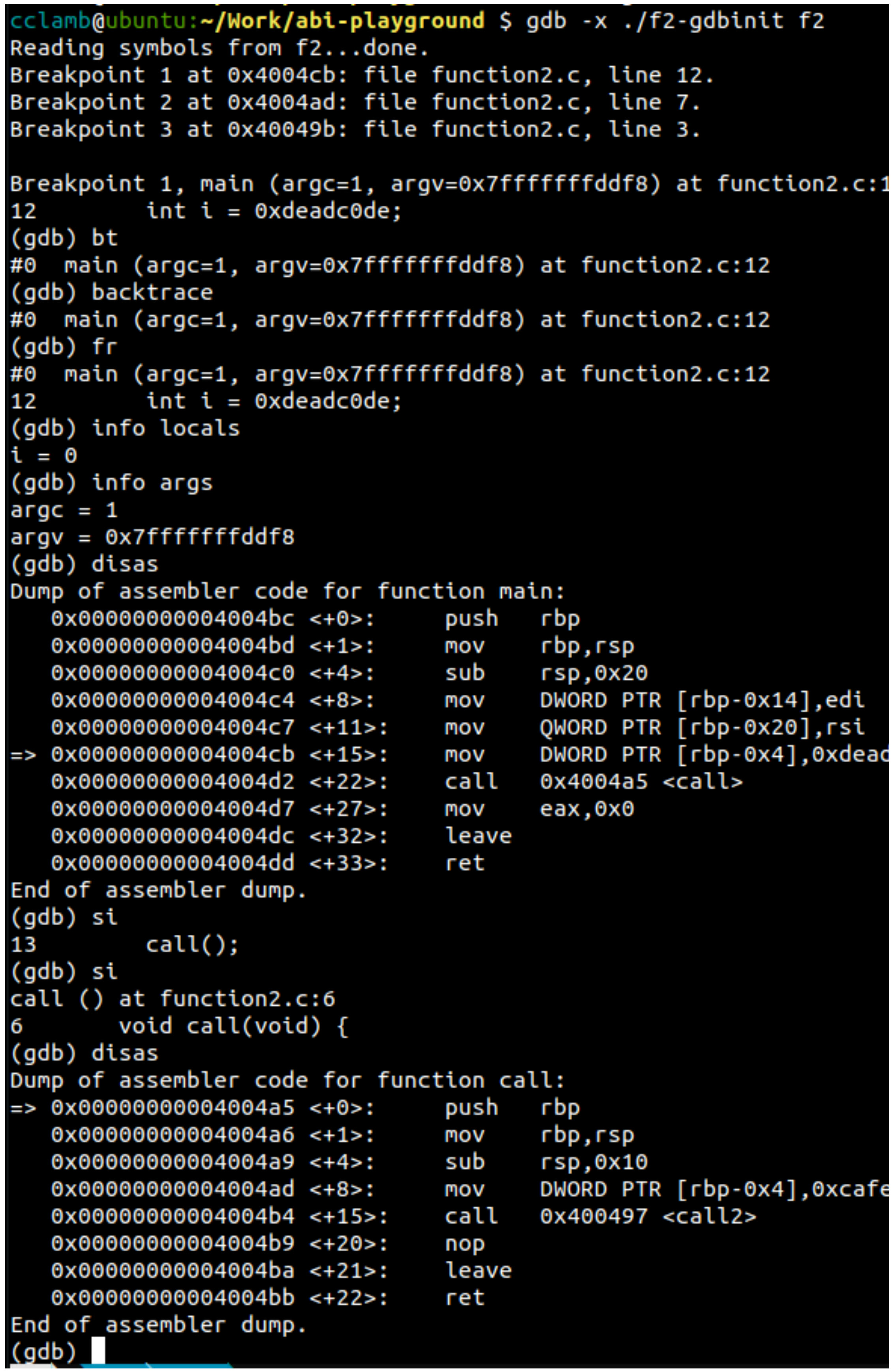
- main, call, call2
- define in gdbinit file
- start GDB

More commands

- info locals, args
- bt -> backtrace
- fr -> frame

What's with RBP and RSP?

- Registers, setting up stack!



```
cclamb@ubuntu:~/Work/abi-playground $ gdb -x ./f2-gdbinit f2
Reading symbols from f2...done.
Breakpoint 1 at 0x4004cb: file function2.c, line 12.
Breakpoint 2 at 0x4004ad: file function2.c, line 7.
Breakpoint 3 at 0x40049b: file function2.c, line 3.

Breakpoint 1, main (argc=1, argv=0x7fffffffddf8) at function2.c:1
12          int i = 0xdeadc0de;
(gdb) bt
#0  main (argc=1, argv=0x7fffffffddf8) at function2.c:12
(gdb) backtrace
#0  main (argc=1, argv=0x7fffffffddf8) at function2.c:12
(gdb) fr
#0  main (argc=1, argv=0x7fffffffddf8) at function2.c:12
12          int i = 0xdeadc0de;
(gdb) info locals
i = 0
(gdb) info args
argc = 1
argv = 0x7fffffffddf8
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004bc <+0>:     push   rbp
   0x00000000004004bd <+1>:     mov    rbp,rsp
   0x00000000004004c0 <+4>:     sub    rsp,0x20
   0x00000000004004c4 <+8>:     mov    DWORD PTR [rbp-0x14],edi
   0x00000000004004c7 <+11>:    mov    QWORD PTR [rbp-0x20],rsi
=> 0x00000000004004cb <+15>:    mov    DWORD PTR [rbp-0x4],0xdead
   0x00000000004004d2 <+22>:    call   0x4004a5 <call>
   0x00000000004004d7 <+27>:    mov    eax,0x0
   0x00000000004004dc <+32>:    leave
   0x00000000004004dd <+33>:    ret
End of assembler dump.
(gdb) si
13          call();
(gdb) si
call () at function2.c:6
6       void call(void) {
(gdb) disas
Dump of assembler code for function call:
=> 0x00000000004004a5 <+0>:     push   rbp
   0x00000000004004a6 <+1>:     mov    rbp,rsp
   0x00000000004004a9 <+4>:     sub    rsp,0x10
   0x00000000004004ad <+8>:     mov    DWORD PTR [rbp-0x4],0xcafe
   0x00000000004004b4 <+15>:    call   0x400497 <call2>
   0x00000000004004b9 <+20>:    nop
   0x00000000004004ba <+21>:    leave
   0x00000000004004bb <+22>:    ret
End of assembler dump.
(gdb) 
```

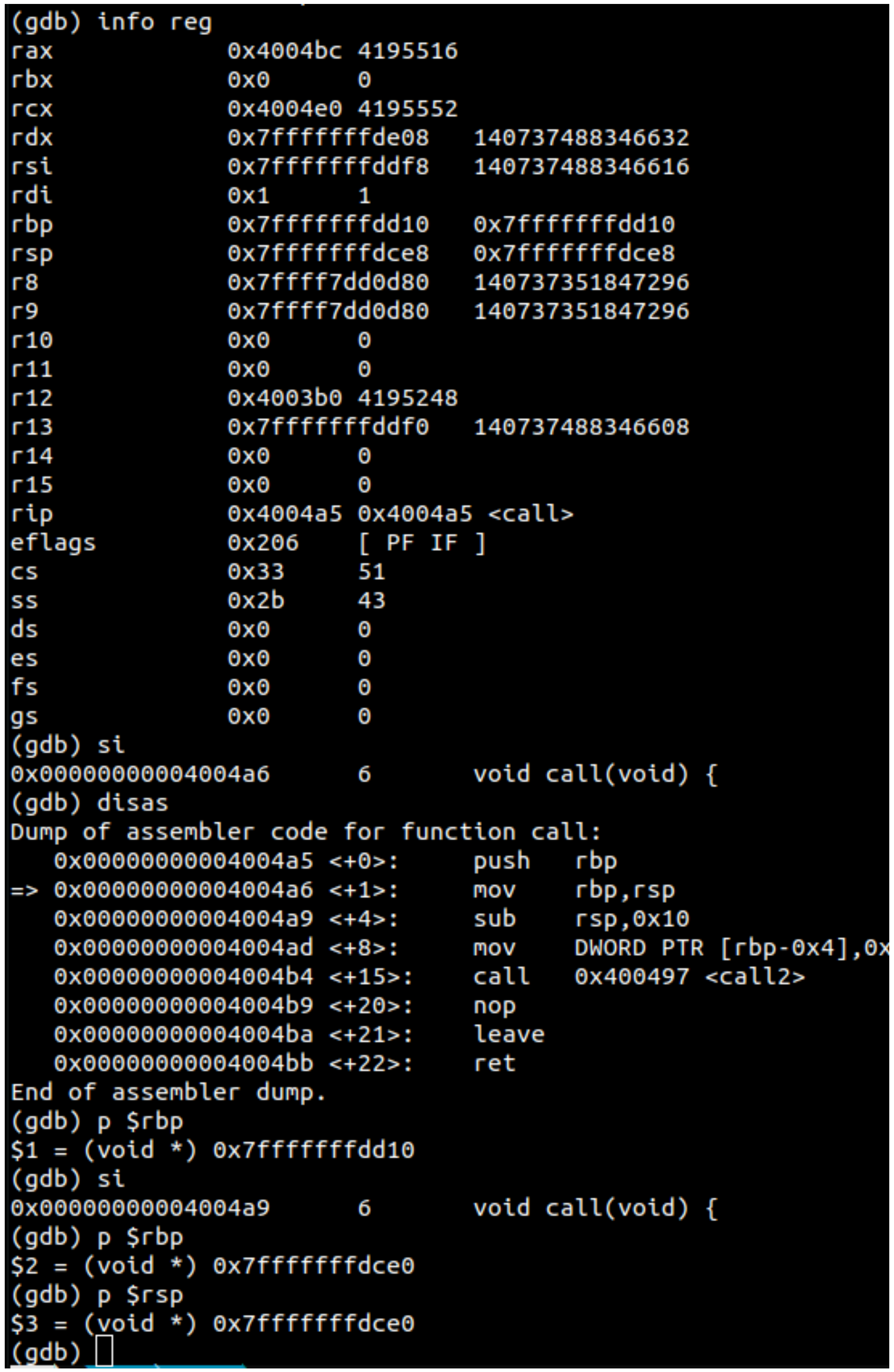
Registers & Memory

REGISTERS

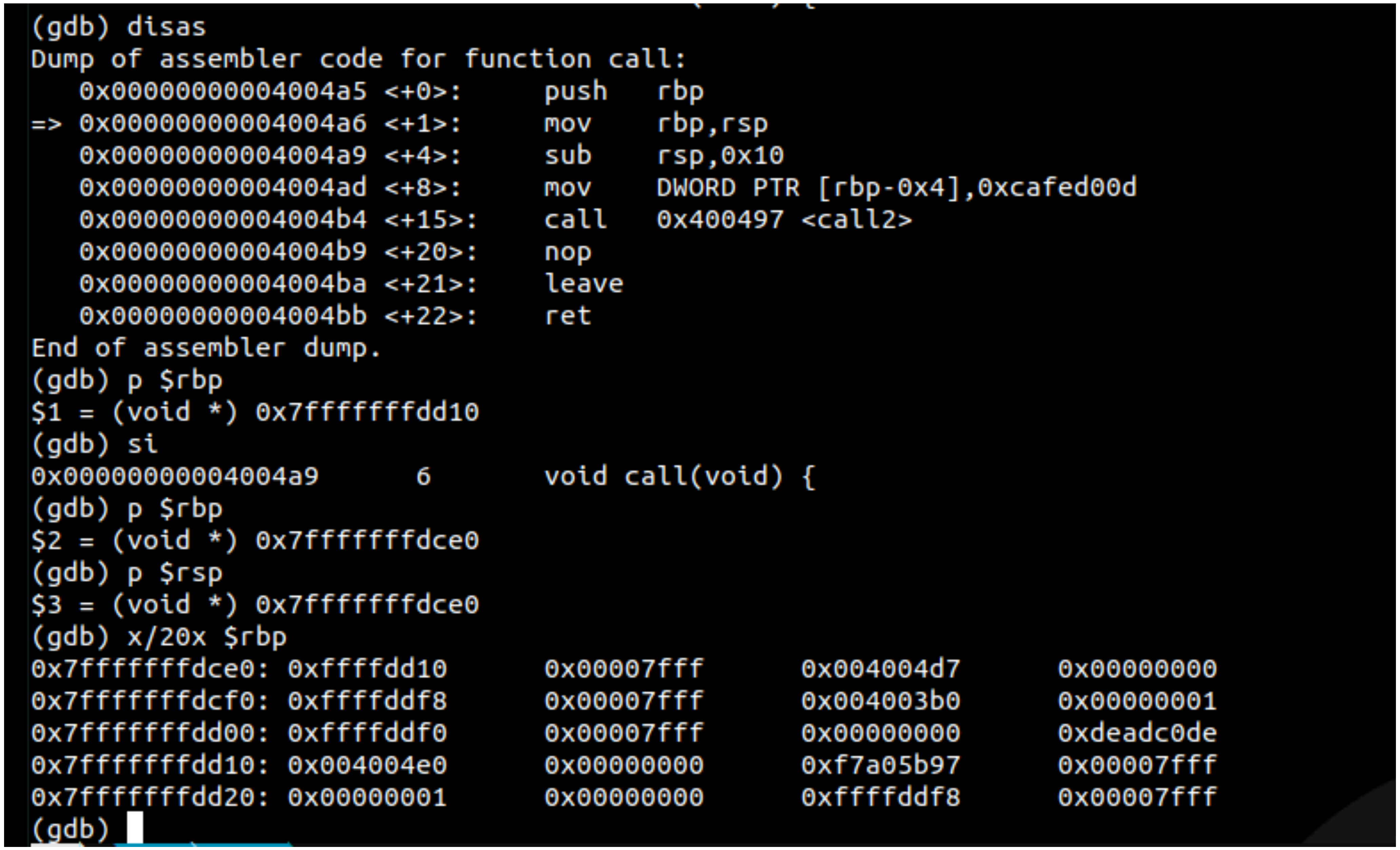
- on-chip, store small bits of data
  - addresses, arithmetic values, control bits
- Very fast access

SPECIAL REGISTERS

- rbp, rsp, rip



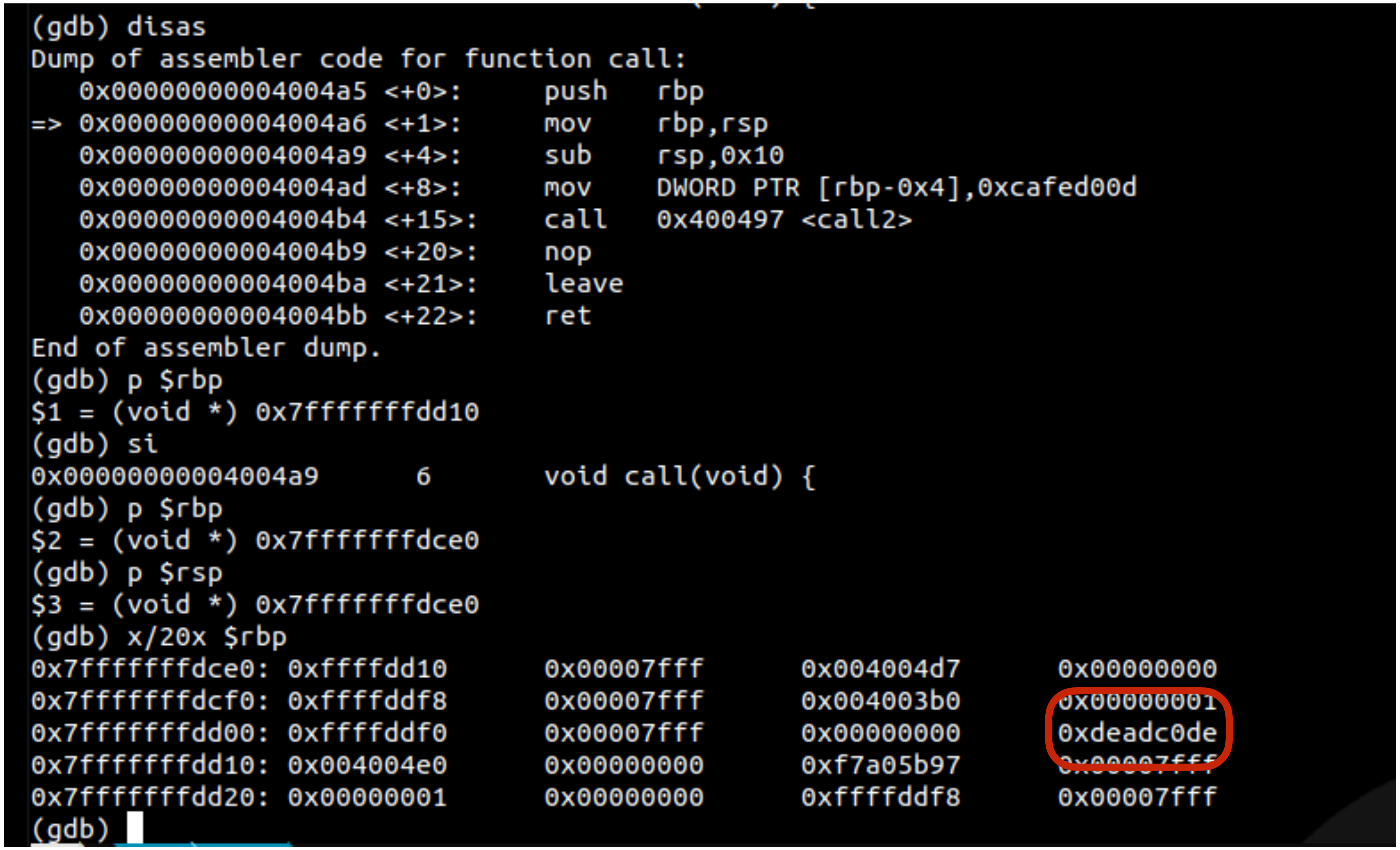
```
(gdb) info reg
rax            0x4004bc 4195516
rbx            0x0      0
rcx            0x4004e0 4195552
rdx            0x7fffffffde08   140737488346632
rsi            0x7fffffffddf8   140737488346616
rdi            0x1      1
rbp            0x7fffffffdd10   0x7fffffffdd10
rsp            0x7fffffffdce8   0x7fffffffdce8
r8             0x7ffff7dd0d80   140737351847296
r9             0x7ffff7dd0d80   140737351847296
r10            0x0      0
r11            0x0      0
r12            0x4003b0 4195248
r13            0x7fffffffddf0   140737488346608
r14            0x0      0
r15            0x0      0
rip            0x4004a5 0x4004a5 <call>
eflags         0x206    [ PF IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb) si
0x00000000004004a6      6       void call(void) {
(gdb) disas
Dump of assembler code for function call:
   0x00000000004004a5 <+0>:     push   rbp
=> 0x00000000004004a6 <+1>:     mov    rbp,rsp
   0x00000000004004a9 <+4>:     sub    rsp,0x10
   0x00000000004004ad <+8>:     mov    DWORD PTR [rbp-0x4],0x
   0x00000000004004b4 <+15>:    call   0x400497 <call2>
   0x00000000004004b9 <+20>:    nop
   0x00000000004004ba <+21>:    leave
   0x00000000004004bb <+22>:    ret
End of assembler dump.
(gdb) p $rbp
$1 = (void *) 0x7fffffffdd10
(gdb) si
0x00000000004004a9      6       void call(void) {
(gdb) p $rbp
$2 = (void *) 0x7fffffffdce0
(gdb) p $rsp
$3 = (void *) 0x7fffffffdce0
(gdb) 
```



```
(gdb) disas
Dump of assembler code for function call:
   0x00000000004004a5 <+0>:     push   rbp
=> 0x00000000004004a6 <+1>:     mov    rbp,rsp
   0x00000000004004a9 <+4>:     sub    rsp,0x10
   0x00000000004004ad <+8>:     mov    DWORD PTR [rbp-0x4],0xcafed00d
   0x00000000004004b4 <+15>:    call   0x400497 <call2>
   0x00000000004004b9 <+20>:    nop
   0x00000000004004ba <+21>:    leave
   0x00000000004004bb <+22>:    ret
End of assembler dump.
(gdb) p $rbp
$1 = (void *) 0x7fffffffdd10
(gdb) si
0x00000000004004a9      6       void call(void) {
(gdb) p $rbp
$2 = (void *) 0x7fffffffdce0
(gdb) p $rsp
$3 = (void *) 0x7fffffffdce0
(gdb) x/20x $rbp
0x7fffffffdce0: 0xffffdd10      0x00007fff      0x004004d7      0x00000000
0x7fffffffdcf0: 0xffffddf8      0x00007fff      0x004003b0      0x00000001
0x7fffffffdd00: 0xffffddf0      0x00007fff      0x00000000      0xdeadc0de
0x7fffffffdd10: 0x004004e0      0x00000000      0xf7a05b97      0x00007fff
0x7fffffffdd20: 0x00000001      0x00000000      0xffffddf8      0x00007fff
(gdb) 
```

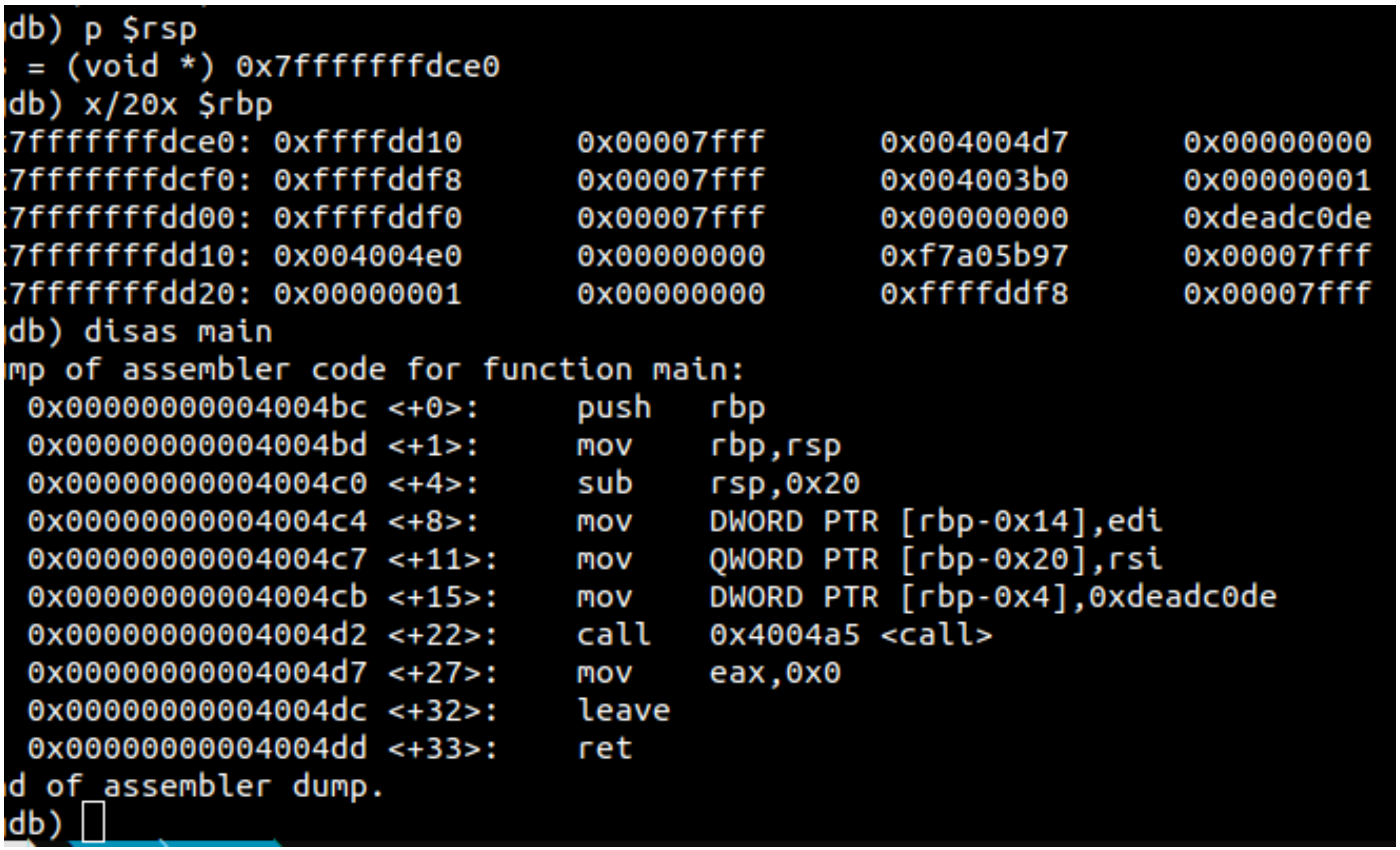
Memory Contents

x/*nfu* address; here **x/20x $rbp** uses hex and rbp content



```
(gdb) disas
Dump of assembler code for function call:
   0x00000000004004a5 <+0>:     push   rbp
=> 0x00000000004004a6 <+1>:     mov    rbp,rsp
   0x00000000004004a9 <+4>:     sub    rsp,0x10
   0x00000000004004ad <+8>:     mov    DWORD PTR [rbp-0x4],0xcafed00d
   0x00000000004004b4 <+15>:    call   0x400497 <call2>
   0x00000000004004b9 <+20>:    nop
   0x00000000004004ba <+21>:    leave
   0x00000000004004bb <+22>:    ret
End of assembler dump.
(gdb) p $rbp
$1 = (void *) 0x7fffffffdd10
(gdb) si
0x00000000004004a9      6       void call(void) {
(gdb) p $rbp
$2 = (void *) 0x7fffffffdce0
(gdb) p $rsp
$3 = (void *) 0x7fffffffdce0
(gdb) x/20x $rbp
0x7fffffffdce0: 0xffffdd10      0x00007fff      0x004004d7      0x00000000
0x7fffffffdcf0: 0xffffddf8      0x00007fff      0x004003b0      0x00000001
0x7fffffffdd00: 0xffffddf0      0x00007fff      0x00000000      0xdeadc0de
0x7fffffffdd10: 0x004004e0      0x00000000      0xf7a05b97      0x00007fff
0x7fffffffdd20: 0x00000001      0x00000000      0xffffddf8      0x00007fff
(gdb)
```

This Looks Familiar!



```
db) p $rsp
 = (void *) 0x7fffffffdce0
db) x/20x $rbp
7fffffffdce0: 0xffffdd10      0x00007fff      0x004004d7      0x00000000
7fffffffdcf0: 0xffffddf8      0x00007fff      0x004003b0      0x00000001
7fffffffdd00: 0xffffddf0      0x00007fff      0x00000000      0xdeadc0de
7fffffffdd10: 0x004004e0      0x00000000      0xf7a05b97      0x00007fff
7fffffffdd20: 0x00000001      0x00000000      0xffffddf8      0x00007fff
db) disas main
mp of assembler code for function main:
 0x00000000004004bc <+0>:     push   rbp
 0x00000000004004bd <+1>:     mov    rbp,rsp
 0x00000000004004c0 <+4>:     sub    rsp,0x20
 0x00000000004004c4 <+8>:     mov    DWORD PTR [rbp-0x14],edi
 0x00000000004004c7 <+11>:    mov    QWORD PTR [rbp-0x20],rsi
 0x00000000004004cb <+15>:    mov    DWORD PTR [rbp-0x4],0xdeadc0de
 0x00000000004004d2 <+22>:    call   0x4004a5 <call>
 0x00000000004004d7 <+27>:    mov    eax,0x0
 0x00000000004004dc <+32>:    leave
 0x00000000004004dd <+33>:    ret
d of assembler dump.
db)
```

main(.)

…and another use of **disas**!

Next, a bit more on these commands.